This application is submitted in the name of the following inventor:

Inventor	Citizenship	Residence Address
Cohen, Earl	United States	43750 Cameron Hills Drive Fremont, CA 94539

The assignee is Cisco Technology, Inc., a California corporation having an office at 170 West Tasman Drive, San Jose, California 95124.

Title of the Invention

Route/Service Processor Scalability Via Flow-Based Distribution of Traffic

Background of the Invention

1. Field of the Invention

The present invention relates to architecture for a router. More particularly, the present invention relates to architecture for a router with scalable processing power.

2. *The Prior Art*

Computers generate and utilize large amounts of data. Different components in a computer network, for example, several different computers each with an internet connection or "uplink", need to be linked together to allow for the transfer of data among the different components in the network. Routers perform this function. Routers take data input from one component in a network and ensure that it is properly transferred to another component in the network.

Different components in a computer network may also generate data in different forms and at different speeds. A router system in conjunction with other network interface components can properly bundle different types of information, transmit the information among different components in a computer network, and ensure that each component of the computer network is allowed to give and receive data at a rate proper for that particular component. Examples of network interface components include Ethernet transceivers and CT3 HDLC channel link data managers (as embodied in the PMC-Sierra CT3 interface chip).

Data bytes are generally bundled into "packets", a discrete grouping of information bytes that will be transferred together along the router system. Packets containing related information in a logical order are grouped into "flows". A flow consists of a unidirectional stream of packets to be transmitted between a particular source and a

particular destination. Packets within the same flow will have the same source/destination address, the same source/destination port, and the same protocol. It is required to maintain the packet ordering within a flow.

Speed and accuracy are two important features of routers. If the data transfer rate, or throughput rate, is not rapid enough, the end user of the required data must wait for it to arrive, wasting valuable processing time. Also, input and output queues waiting to transfer data may overflow if information is being added more rapidly than it is being removed. Accuracy in transfer is also important to ensure that data arrives at its proper destination in the proper order.

Routers can also provide services as they perform the data transfer or switching. Some of these services involve gathering information about the data being transferred or performing some other processing function upon the data as it is being transferred from one component in the computer network to another. Some of the more recent service developments involve "touching", or performing some processing function, on the data in most or every packet as it is transferred. Such "high touch" services can require the router system to have a great deal of processing power. New services are being developed all the time, and thus the amount of processing that performing services requires is unbounded.

Current routing systems generally operate on a "one processor per line card" model. A line card is a group of components in a computer network such as a group of modems or an internet connection which sends and receives data. In a "one processor per line card" model, all of the processing for a single line, or a group of related lines, is done by a dedicated processor. This model works acceptably well if the line's rate of data input, or input interface rate, is low enough so that the processor does not become a bottleneck in the system. With low enough interface rates, one processor can handle multiple lines.

However, the line's interface rate may increase as higher-speed computers or internet connections are added. Furthermore, additional router services may become desirable. Eventually, the dedicated processor may no longer be able to keep up with the processing requirements of the line and the throughput rate will slow down. If the throughput rate becomes unacceptably slow, the only solution is generally to replace the dedicated processor with a more powerful machine. This is an expensive solution. In certain cases, for example a high-speed internet line or "uplink", a processor may not even be available that can keep up with the interface rate of the high-speed line. Current internet uplink rates such as Gigabit Ethernet (1 gigabit per second) are already becoming difficult to handle with a single dedicated processor.

Accordingly, it would be desirable to provide a router architecture that would be easily scalable to accommodate future router service growth as well as expand-

able uplink connections, and which would allow processors to be added as more processing power became necessary, without requiring the entire system's future hardware needs to be purchased initially. It would also be advantageous to maintain packet ordering within each flow while still flexibly routing among different processors as necessary.

Summary of the Invention

The invention provides a system and a method for a router architecture that is scalable, that is, as more processing power is desired, more individual processors can be added. The data flow from each line can be distributed among all of the processors in the system. As desired services are added, increasing the amount of "touch" or processing performed on the packets in the system, more processors can be added to carry the increased load. The router architecture is also able to distribute the high interface rate of an uplink connection in the same manner.

In a preferred embodiment, packets are allocated to processors in a manner that allows the original order of data packets within the same flow to be maintained. The system uses a hash function to distribute the flows, making sure that packets within the same flow are sent to the same processor so that the original packet order in each flow is maintained. Different flows may be sent to different processors.

Brief Description of the Drawings

Figure 1 shows a logical block diagram of the router architecture of the present invention.

Figure 2 shows a sample flow diagram for a data packet.

Figure 3a shows an example port adapter design for an Ethernet interface.

Figure 3b shows an example port adapter design for a channelized T3 line (CT3) interface.

Figure 4 shows an example Data Compiler.

Figure 5 shows an example packet digest format.

Detailed Description of a Preferred Embodiment

In the following description, a preferred embodiment of the invention is described with regard to preferred process steps and data structures. However, those skilled in the art would recognize, after perusal of this application, that embodiments of the invention may be implemented using a set of general purpose computers operating

under program control, and that modification of a set of general purpose computers to implement the process steps and data structures described herein would not require undue invention.

The present invention provides a router architecture capable of implementing a wide variety of services while balancing the router system load among a number of processors. The router architecture of the present invention is also capable of distributing the load from a single uplink among the multiple processors. The ability to distribute the router system load among all of the processors in the system makes the system throughput scalable; the throughput rate increases with each additional processor. Because there is no cooperation problem among the processors and each is acting independently and preferably on separate flows, the system throughput is expected to scale approximately linearly with the number of processors.

The use of multiple processors which are redundant also provides added reliability. If one processor fails, the processing tasks can be allocated among the remaining processors.

FIG. 1 shows a logical block diagram of the components in the router system. FIG. 1 is intended to show the interconnections and resulting scalability of the router system architecture, and is not meant to be an accurate representation of the number of the various components in the router system in any particular embodiment. Indi-

vidual data interface components 10 are connected to a set of Port Adapters ("PA"s) 20, which are in turn connected to a set of Data Compilers (DCs) 30. The DCs 30, an Uplink 60, and a set of Route Processing Engines (RPEs) 50 are all interconnected through an Xbar 40. The Uplink 60 is connected to an external network such as the internet 70. Data transfer originates at either the individual data interface components 10 or the internet 70 and travels through the router system to the Xbar 40, where it is sent on to be processed by a selected one of the RPEs 50. Once processing is complete, the data is transferred back through the Xbar 40 to its final destination, either the internet 70 or an individual data interface component 10.

Data flow through the system is more easily followed through a sample flow diagram for a data packet as shown in FIG. 2. FIG. 2 shows only one example of a data packet flow path; numerous other paths are possible using a router system of the present invention. FIG. 2 shows an individual computer generating the original data. This data may be sent out via modem to be routed to another part of the router system. Methods of transferring data via modem are well known in the art of computer networking systems. The example in FIG. 2 of an individual computer and a modem is purely illustrative of an individual data interface component 10 in a router system as shown in FIG. 1. Other types of data interface components could serve the same function; for example, a T1 connection (1.5 Mbps) could also function to provide data to a PA 20.

In step 110, the data is first sent to the PA 20. A single PA 20 is capable of handling the throughput from multiple modems or other individual data interface components 10 since such components send data at a relatively slow and intermittent rate. The PA 20 bundles this data it has received from multiple modems or other interface lines into packets. Each packet is represented as a series of descriptors in a format compatible with the PA 20. Network interface components, such as many Ethernet transceivers and HDLC channel managers, use a descriptor approach to transfer packet data to a local memory, possibly with the help of a local microprocessor. Each descriptor effectively indicates a memory block (called a particle) and a size, as well as control information such as whether this is the first or last descriptor in a packet. These packets are the main units of data sent across computer network systems. The creation and use of data packets are well known in the art of computer networking systems.

Various types of PAs are available to convert between different user interfaces and a generic PCI bus. The PCI (Peripheral Component Interconnect) bus is known in both the computer and networking industries. FIG. 3a shows a representative PA design for an Ethernet interface. FIG. 3b shows a representative PA design for a channelized T3 line (CT3) interface. Other user interfaces such as OC-3 (a fiber-optic standard) may also be converted using a PA.

In step 112, the data packets from the PA 20 are sent to a DC 30. A DC provides an interface between the internal fabric in the router and a generic PCI bus

which can connect to a PA. Figure 4 shows a representative Data Compiler. The DC 30 further prepares the packets for transfer along the routing system by putting them into packet digest form, and then forwards the packets along to the Xbar 40. A packet digest contains both a header and a payload. The DC 30 will further "packetize" the incoming packets by converting them from the particles to a consecutive block of memory, forming the "payload" for a packet digest. The DC 30 will also add a packet digest header.

The packet digest header contains information about the packet to aid in the prompt forwarding and processing of the packet, such as—the Xbar 40 source port number, the size of the packet digest header and payload, the destination RPE 50, the packet priority, the protocol type, and the class of service required.

Figure 5 shows an example packet digest format 200. In a preferred embodiment, the packet digest includes the following fields:

- a CMD field 202 to control features of the packet digest (such as the TIMESTAMP field, as described below), and to indicate a type of packet digest (where there is more than one type, such as in alternative embodiments);

- a SRC_PORT field 204 to indicate a source ID of the transmitting entity;

- a PKT_OFF field 205 and a NEW_PKT_OFF field 207 to indicate an number of bytes of padding between the packet digest header and the packet digest payload (where the PKT_OFF in-

dicates a total size of the packet digest header plus any padding, while the NEW_PKT_OFF field indicates a new value for PKT_OFF);

- a PDH_SIZE field 210 to indicate the size of the packet digest header;

- a DEST_OFF field 212 to indicate an offset of the destination IP address in the packet digest header;

- a "FIB Leaf" field 214 to indicate a pointer to the leaf in a FIB lookup m-trie for the destination IP address;

- a DEST_MASK field 216 to indicate which Xbar ports are intended as destinations for the packet (where a selected bit <39> of the DEST_MASK field preferably indicates a multicast packet);

- a QUEUE field 218 to indicate to which hardware output queue at the particular output interface to direct the packet;

- an INTERFACE field 220 to indicate to which hardware output interface to direct the packet;

- a TIMESTAMP field 222 to indicate a timestamp or other data to be written by the inbound Xbar interface (where a selected bit CMD <0> indicates that the TIMESTAMP field should be written to); and

- o a set of additional words reserved for software classification data.

In order to determine the destination RPE 50, the DC 30 includes a classification engine coupled to the crossbar interface to determine which RPE 50 will receive each packet. Packets are not distributed randomly among RPEs, so as to maintain the original ordering of packets within the same flow. The classification engine distributes packets so that packets within the same flow will be sent to the same RPE, but packets in different flows may be sent to different RPEs and possibly reordered as determined by RPE resources.

The classification engine analyzes the router system traffic and distributes flows to multiple processors. The distribution is partly table-driven, and is thus flexible and easily changeable, even dynamically. In a preferred embodiment, the classification uses a hash function of packet flow information to select a specific RPE 50 to handle processing for that packet flow. The hash function distributes packets evenly among the processors in response to flow information such as the source/destination address, the source/destination port, and the protocol. The hash function can operate using any information that will allow for flow preservation.

In a preferred embodiment, the hash function performs an XOR (exclusive OR) logical function of several bytes of fields from the IP header of the packet, including the following fields:

- a source IP address (four bytes);
- a destination IP address (four bytes); and
- a protocol type value (one byte).

The XOR logical function is performed for all nine bytes, providing a single byte output value, which can be used to index into a 256-entry table for selecting the RPE 50.

Once the DC 30 has determined the packet's destination RPE 50 and the packet header and payload are ready to be sent out, the packet (now in packet digest form) is placed in an input queue to be stored until it is sent to the selected RPE 50. Note that the amount of processing performed by the DC 30 on each packet is bounded; the DC 30 only does a fixed amount of work per packet. There is no cost in terms of DC 30 processing for adding additional services, as all such additional processing will be performed by the RPEs 50. Thus, the DC 30 is prevented from becoming a system bottleneck as more services are added. In the router system of the present invention, DC 30

input queues should only overflow if there are not enough RPEs 50 to support the requested level of services at the incoming data rates; if input queues overflow, it is always possible to add more RPEs 50.

In step 114, packets are sent from the DC 30 to the Xbar 40. The Xbar 40 provides the interconnection between the data origination component ports, the router system processor ports, and the data destination component ports. In one data transfer, the data packet will cross through the Xbar 40 twice (step 114 and step 118 in FIG. 2). The Xbar 40 receives an incoming packet and transfers it to the selected RPE, and then the Xbar 40 transfers the packet back to its destination port. The Xbar 40 must provide enough bandwidth to handle the data interface rates of both the DCs 30 and the Uplink 60. Xbar interconnections are known in the art of computer and router systems.

The Xbar 40 includes a number of ports; each port is a pair of unidirectional links. The Xbar 40 implements a simple arbitration scheme between sending and receiving components, ignoring whether the destination port has the resources to handle the packet or not. Packets can be dropped at any point if there are not sufficient resources to handle them. Software must ensure that there are sufficient resources available at a destination port, for example, memory space at the output to queue the packet, before sending a packet over the Xbar 40. Output queuing protocols determine what occurs when an output queue overflows; output queue management is performed at the DC 30.

The Xbar 40 includes an arbiter — ports forward their requests to the arbiter, and grants are sent back. The Xbar 40 arbiter uses a windowing scheme in which each port can make requests for multiple destinations at the same time; the arbiter attempts to find the best fit between requesting ports and destinations. The Xbar 40 can also support multicast — a port can request that a packet be sent to multiple destinations. In the preferred embodiment, the Xbar 40 waits for all ports to be free, and then sends the multicast request all at once. Numerous other Xbar multicast schemes are well known in the art.

The Xbar 40 in the preferred embodiment discussed herein is packet-based, but the Xbar 40 could be designed to be cell-based and still fall within the inventive concepts disclosed herein. It is also possible to provide multiple Xbars or a bus that performs the same function as the Xbar. The methods discussed herein for the implementation of such an Xbar device are by no means limiting. Alternate methods for the implementation of such an Xbar device will readily suggest themselves to those of ordinary skill in the art.

In step 116, the Xbar 40 transfers the packet to the destination RPE 50. There are multiple RPEs 50 connected to the Xbar 40 that provide both switching and services, including the high-touch type of services that may require a great deal of processing power. Such services may also be added later to the system as more services are

developed and become available. Additional RPEs 50 can be added to the router system to provide additional processing power; this provides the scalability feature of the present invention.

An RPE 50 receives a packet digest with a packet digest header containing classification information provided by the DC 30. Based on the classification information, the packet is put into one of a small number of input queues on the RPE 50. The intent of the RPE 50 queuing is to provide the capability for handling both high- and low-priority packets. Packets are processed from among the RPE's input queues by the RPE 50 processor in whatever order it sees fit.

The RPE 50 processor will perform various tasks on the packet, including switching, tag application/update, access list processing (filtering), and all other router service functions. Some of these operations result in the RPE 50 modifying the packet digest, others only update RPE-internal data structures. When the RPE 50 processor is done with a packet, the RPE 50 will know the packet's final destination. The example destination in FIG. 2 is the internet 70 via the Uplink connection 60. However, it would be evident to one of ordinary skill in the art that the packet's destination could also be an individual data interface component 10 via a DC 30 and a PA 20.

In step 118, the packet is put back on the Xbar 40 from the RPE 50. In step 120, the Xbar 40 transmits the packet to the Uplink 60. The Uplink 60 is a high-

bandwidth PA with its own DC. Like a DC, the Uplink 60 is connected to the Xbar 40 via an Xbar link. However, in order to meet its bandwidth requirements, an Uplink 60 may require multiple Xbar ports. The Xbar 40 can support multiple Uplink 60 connections.

In step 122, the Uplink 60 transfers the data to the internet 70, completing the data transfer from the individual data interface component 10 to the internet 70.

The Uplink(s) 60 can also send data to the Xbar 40 for transmission to an RPE 50 and eventual transfer to an individual data interface component 10. The Uplink(s) 60's traffic from the internet 70 is also distributed via the Xbar 40 to the various RPEs 50 using a hash function. In this way, the router system can support a plurality of Uplink(s), each of arbitrary speed.

Various alternate embodiments of the present invention have also been conceived, and would be clear to those skilled in the art after reviewing this application. For example, in alternative embodiments, the DC and a specific PA type can be integrated together to produce a more cost-effective, though less easily adaptable, interface.

Other alternative embodiments include performing some services of a bounded nature on the DCs, as is currently done with output queuing. For example, the DCs could accumulate statistics and do flow-based accounting on each packet. In this

way, the DCs could handle some portion of the known processing load leaving unbounded and future services to the RPEs.

Yet other alternative embodiments include adding specialized processing engines connected to the Xbar. Packets can be sent through these specialized engines either before or after or instead of an RPE to perform services such as compression/ decompression, encryption, or routing. The classification engine on a DC could determine the type of service required by a packet and route it appropriately.

While embodiments and applications of this invention have been shown and described, it would be apparent to those skilled in the art that many more modifications than those mentioned above are possible without departing from the inventive concepts herein. The invention, therefore, is not to be restricted except in the spirit of the appended claims.

91. The method as in claim 90 further comprising:
processing compression as said type of service.

92. The router as in claim 90, further comprising:
processing decompression as said type of service.

93. The router as in claim 90, further comprising:
processing encryption as said type of service.

94. The router as in claim 90, further comprising:
processing routing as said type of service.

95. A computer readable media, comprising:
said computer readable media containing instructions for execution in a processor for the practice of the method of claim 17 or claim 71 or claim 90.

96. Electromagnetic signals propagating on a computer network, comprising:
said electromagnetic signals carrying instructions for execution on a processor for the practice of the method of claim 17 or claim 71 or claim 90.